JavaScript Closures

Stephen P Levitt

School of Electrical and Information Engineering
University of the Witwatersrand

Outline

Read the code and answer the following questions

```
1  const sayHello = function (name) {
2    let text = 'Hello ' + name
3    let say = function print () { console.log(text) }
4    return say
5  }
6
7  const greet = sayHello('Thabo')
8  greet()
9
10 // Output: Hello Thabo
```

- Identify the local variables for `sayHello`
- When do these variables go out of scope?
- What is returned from `sayHello`? Use the correct term.
- At what point is the code in the `print` function executed?
- Explain the output.

> *Closure is when a function can remember and access its lexical scope even when it's invoked outside its lexical scope.*
>
> — Kyle Simpson in *You Don't Know JS*

> *A closure is the combination of a function and the lexical environment within which that function was declared.*

All variables in the outer function form part of the closure

```
function sayAlice () {
  const sayAlert = function greeting () { console.log(alice) }
  // Local variable is hoisted and ends up within closure
  const alice = 'Hello Alice'
  return sayAlert
}

sayAlice()() // immediately invoke the returned function expression

// Output: Hello Alice
```

Modifying variables within the closure

```
function say5 () {
  let num = 5
  const say = function () { console.log(num) }
  num++
  return say
}

say5()() // immediately invoke the returned function expression

// What is the output?
```

With each new call of the outer function a new closure is created

```
const namer = function (name) {
  return function (obj) {
    obj.name = name
    console.log(obj)
  }
}

let anObj = { groupNum: 12 }

const nameFrancis = namer('Francis')
const nameRyan = namer('Ryan')

nameFrancis(anObj)   // name set to Francis
nameRyan(anObj)      // name changed to Ryan

// What is the output?
```

Closures Share Variables

```
let gPrintNumber, gIncreaseNumber, gSetNumber // globals

function setupSomeGlobals () {
  let num = 5
  // Store references to functions through global variables
  gPrintNumber = function () { console.log(num) }
  gIncreaseNumber = function () { num++ }
  gSetNumber = function (x) { num = x }
}

setupSomeGlobals()

gPrintNumber()
gIncreaseNumber()
gPrintNumber()
gSetNumber(44)
gPrintNumber()
```

Closures — Recap

- Inner functions have closure over their lexical scope
- All variables in the outer function form part of the closure
- The scope closed over has the state resulting from the completion of the outer function
- With each new call of the outer function a new closure is created
- All inner functions share access to the same variables

Give the output produced by the following code

```
function itemList () {
  let items = []
  let i = 0
  while (i < 10) {
    let item = function () {
      console.log(i) // should show its number
    }
    items.push(item)
    i++
  }

  return items
}

let list = itemList()
list[0]()
list[5]()
```